1. **What is meant by Spring Cloud?**

   Spring Cloud is a spring module that provides the RAD (Rapid Application Development) feature to the spring framework.

   Spring Cloud is a framework for building robust cloud applications. Spring Cloud provides a solution to the commonly encountered patterns when developing a distributed system.

2. **Why we required Spring Cloud?**

   Spring Cloud framework provides tools for developers to build a robust cloud application quickly. We can also build the micro service-based applications.

   For example, configuration management, service discovery, circuit breakers, intelligent routing.

3. **Explain the features of Spring Cloud?**

   The great part about Spring Cloud is that it builds the concept of Spring Boot. Spring cloud is built upon some of the common building blocks of spring framework which are as follows:

   - ✓ Intelligent routing and service discovery
   - ✓ Service-to-Service Call
   - ✓ Load Balancing
   - ✓ Leadership Election
   - ✓ Distributed Configuration
   - ✓ Distributed Messaging

4. **Explain the Spring Cloud components?**
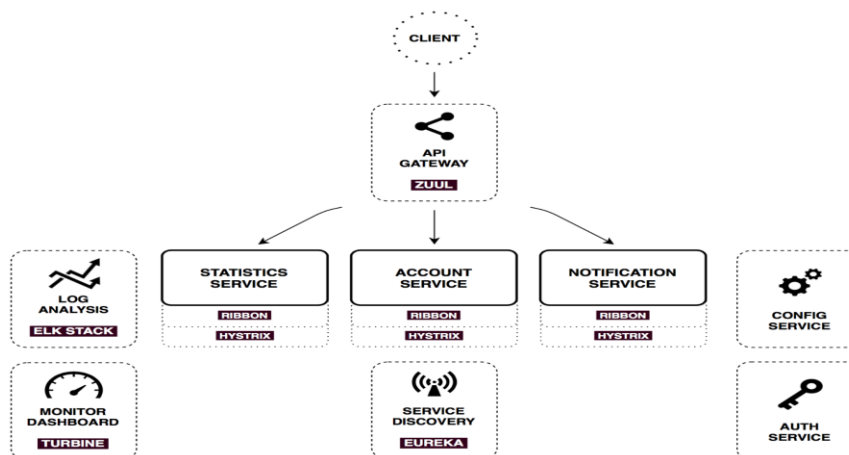
Configuration: Spring Cloud configuration components provide server-side and client-side support for externalized configuration in a distributed system. We can manage the external properties with config server for applications across all environments. Spring Cloud config server can use Git, SVN (Apache Subversion), filesystem, and Vault to Store config.

Config clients (microservice app) retrieve the configuration client from the server on startup.
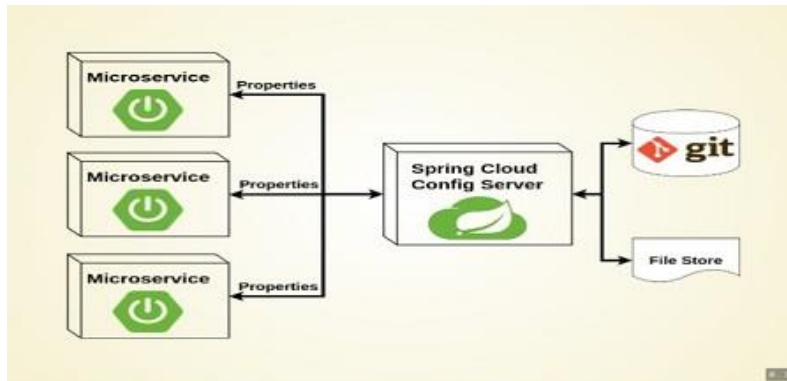
Service Discovery (Eureka Naming Server): Eureka naming server is an application that holds information about all client service applications. Each microservice registers itself with the Eureka naming server. The naming server registers the client services with their port numbers and IP addresses. It is also known as Discovery Server. Eureka naming server comes with the bundle of Spring Cloud. It runs on the default port 8761.

There are two discovery patterns: Client-side discovery and Server-side discovery.

Client-side discovery: In the Client-side discovery, client is responsible for determining the network location of available services. The client uses a load-balancing algorithm to select one of the available services and make a request. Netflix OSS is an example of a client-side discovery pattern.

Server-side discovery: In the server-side discovery, the client makes an HTTP request to a service through a load balancer. The load balancer contacts to service registry and route each request to an available service instance. Similar to client-side discovery, service instances are registered and deregistered with the service registry.

Circuit Breakers: Netflix has created a library called Hystrix. It implements the circuit breakers pattern. Circuit breakers calculate when to open and close the circuit and what to do in case of failure. When all services fail at some point, the circuit breaker handles these failures gracefully. The circuit breakers have three states: OPEN, CLOSED, and HALF-OPEN State.

OPEN State: Application flow is working like below. Client calling the supplier in middle Circuit-Breaker sits,

The circuit breaker returns an error calls without executing the supplier api.

CLOSED State: Application flow is working like below. Client calling the supplier in middle Circuit-Breaker sits,

In the Above flow if Circuit-Breaker is in closed state all calls pass through to the supplier micro services. It responds without any latency.

HALF-OPEN State: Application flow is working like below. Client calling the supplier in middle Circuit-Breaker sits,

The circuit turns to HALF-OPEN state when a function execution is timed out. It test that underlying problem still exists or not. It is a monitoring and feedback mechanism. It makes a trial call to supplier micro services to check if it has recovered. If the call to the supplier is timed out, then the circuit remains in the OPEN state. If the call return success, the circuit-switched to the CLOSED state. The circuit breaker returns all external calls to the service with an error during the HALF-OPEN State

Routing: The cloud application made up of many micro services so the communication will be critical. Spring Cloud supports communication via messaging or HTTP request. Routing uses Netflix Ribbon and Open Feign.

API Gate Way: API Gateway allows us to route API request (external or internal) to connect services. It also provides a library for building an API gateway on the top of Spring MVC. Its aims to provide cross-cutting concerns to them, such as security and monitoring

Tracing: Spring Cloud's other functionality is distributed tracing. Tracing is a single request to get data from the application. Tracing results in an exponentially larger number of requests to various micro services.

We can add Spring Cloud Sleuth library in our project to enable tracing. Sleuth is responsible for recording timing, which is used for latency analysis. We can export this timing to Zipkin.

5. **What is the use of Spring Cloud Bus?**

   It is a message broker in micro services environment will use to update the files defined in github without restarting the application.

6. **What is Pivotal Cloud Foundry(PCF)?**

   Cloud Foundry is an open-source, multi-cloud Platform as a Service (PaaS). You can deploy your application on your own computing infrastructure.
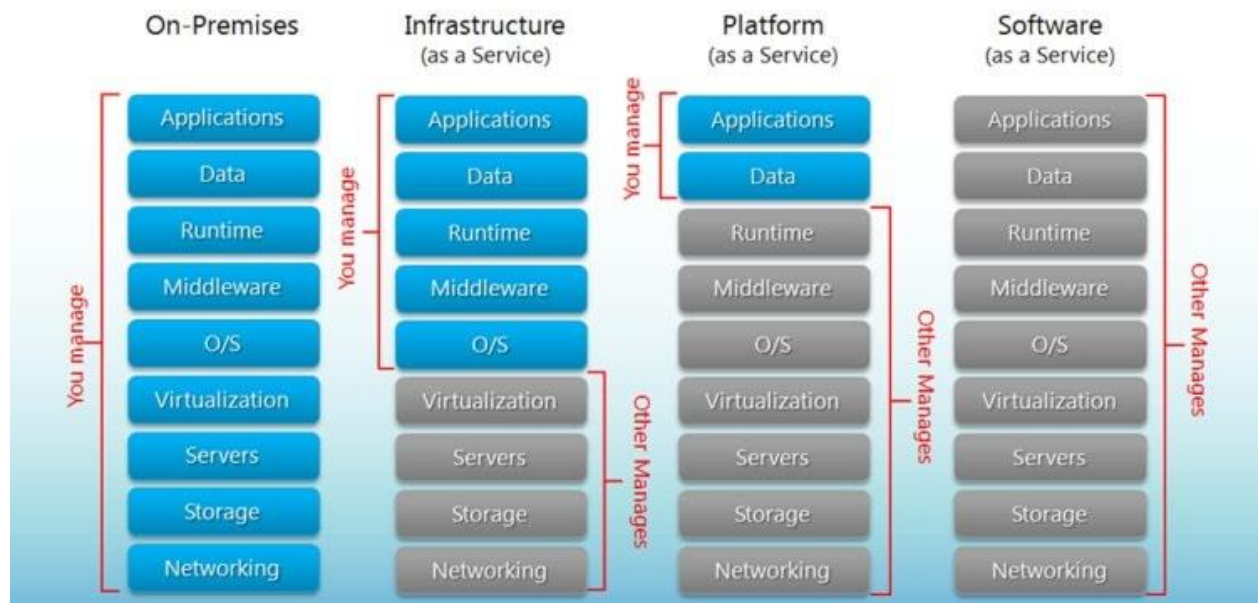
7. **Differences between On-Premise, SaaS, PaaS, IaaS?**

   IaaS: cloud-based services, pay-as-you-go for services such as storage, networking, and virtualization.
   PaaS: hardware and software tools available over the internet.
   SaaS: software that's available via a third-party over the internet.
   On-premise: software that's installed in the same building as your business.

8. **What is the major difference between Spring Cloud and Spring Boot?**

   Spring cloud is a microservice management and coordination framework used to integrate and manage individual microservices while Spring boot is used to develop these microservices.